平成 12 年度

学士学位論文

形状適応 DCT 処理の データ駆動型並列実現法

A Data-Driven Implementation of Shape Adaptive DCT

1010431　　橋本 正和

指導教員　　岩田 誠

2001 年 2 月 5 日

高知工科大学 情報システム工学科

要 旨

形状適応 DCT 処理の
データ駆動型並列実現法

橋本 正和

近年、MPEG や H261,H263 などの画像圧縮伸張方式が多数標準化されている。また、映像処理向き LSI プロセッサとして、データ駆動型マルチメディアプロセッサ DDMP の性能を確認している [1]。DDMP はプログラム可能で、MPEG4 の多様なアプリケーションツールに柔軟に対応できる。

本研究は、この DDMP を MPEG4 向きに最適化することを目標に、圧縮伸張処理中で高負荷となる SA-DCT 処理の、データ駆動型並列実現法を提案する。SA-DCT は普通の DCT 処理に、オブジェクトのシフト操作が加わる。SA-DCT 処理においての並列性を利用して、パイプライン並列にプログラム化した場合、DDMP 内の問題点を解決するための最適化を行う。そのために、SA-DCT 処理のボトルネックとなる部分の検索と、解消法を検証する。その方法として、新命令の追加による高速化を示し、その実現を、簡単なハードウェア機構の追加によって可能とした。命令は汎用性を持たせたものとし、幅広い分野への応用が期待できる。本稿では、簡単なハードウェアの見積もりを行った。

キーワード　MPEG4, DDMP, データ駆動, SA-DCT, 並列処理, 命令 ～

Abstract

A Data-Driven Implementation of Shape Adaptive DCT

MASAKAZU HASHIMOTO

In recent years, picture compression and extension systems, such as MPEG, H261 and H263, etc., have become standards. Also, We are improving the performance of DDMP (Data Driven Multimedia Processor) as an one chip processor for image processing[1]. DDMP is a programmable device and it has enough flexibility for the various application tools of MPEG4.

This research proposes a method of data driven type parallel processing for SA-DCT which is heavy tasks in image one of compression and extension the good of this paper is optimizing DDMP to MPEG4 functions. Shift operations of pixels are required by SA-DCT additional ordinal DCT. The method solves some problems in the current DDMP instruction set to utilize the parallelism of SA-DCT. Therefore, the search and examination on the bottleneck of SA-DCT and its solution is urgently required. The improvement in the speed by addition of the new instruction is proposed, and the possibility of realizations shown by a description of simple hardware mechanisms. The new added instructions should give more flexibility and can be applied in various fields of application. In this paper, a rough estimation of hardware cost is describes.

key words MPEG4, DDMP, data-driven, SA-DCT, parallel processing, instruction

目次

図目次

表目次

第 1 章

序論

近年、MPEG や H261,H263 などの画像圧縮伸張方式が多数標準化され、インターネットや TV 電話、DVD、デジタルカメラ等、様々なメディアで使われるようになってきている。また、携帯端末を始めとした移動体情報機器に、映像情報を介した通信サービスを提供する機運が高まっている。

MPEG とは ISO/IEO の動画像音声符号化グループ (Moving Picture coding Expert Group) のことであり、1988 年から動画と音声のデジタル圧縮符号化方式の規格化に取り組み、現在までに MPEG1、MPEG2、MPEG4 が制定されている [2][3]。

MPEG1 は CD-ROM 等の蓄積メディア用に開発され、MPEG2 は放送分野に応用することを目的としている。

これに対して MPEG4 は、より帯域の狭い携帯端末や TV 電話を対象とし、また、マルチメディアデータを自由に扱うための汎用符号化標準として、超高能率符号化方式や、伝送誤りに強い符号化方式を採用している。MPEG4 の最も大きな特徴は、多様なアプリケーションをサポートするために、様々な要素技術を組み合わせたプロファイルとレベルを複数提供している点である。

現在、MPEG4 をサポートした機器がいくつか商標化されているが、それらはシンプルプロファイルのみであり、MPEG4version2 と呼ばれる高機能なプロファイルのサポートが必要になる。

MPEG4version2 では、受信端末で様々な操作も可能な符号化方式として、オブジェクト符号化方式を採用しており、符号化する 3 次元の情報を空間内の各オブジェクトに効率の良い符号化をし、符号化効率を高めている。また MPEG4 では従来の自然映像や自然オーディ

オ信号に加えて、作成時にオブジェクトを分離しやすいコンピュータグラフィックスデータや、合成音声・音響信号も扱えるようになっており、オブジェクト符号化が特に有効であると期待できる。

MPEG4 の映像符号化方式は、MPEG1,2 と同様、動き補償予測および DCT(Discrete Cosine Transform) の方式に基づいている。動き補償予測とは、動画像で動いている部分を検出し、この検出部分と 1 つ前の画面から次の画面を予測することである。DCT は、映像信号のうち視覚的に重要でない部分の情報を削るために、明るさの変化の少ない成分と変化の大きな成分に変換する手法である。自然動画像はオブジェクトごとに符号化されるため、従来の符号化方式とは異なり、動きを表す情報 (動き推定) と画面の模様を表すテキスチャ情報のほかに、オブジェクトの形状そのものを符号化 (形状情報符号化) する必要がある。このように、オブジェクト単位で適用する DCT を Shape Adaptive DCT (SA-DCT) と呼ぶ。

しかし、この SA-DCT を専用ハードで実現しようとすると、個々のプロファイルやレベルにより、ハードを一から作り直さなければならない。

そこで本研究では、これらのプロファイルやレベルを、処理データに応じて、ソフトウェアにより柔軟に設定できる端末を、十分な実行速度で動作させ、極省電力化するための超並列実行可能な、携帯機器向き MPEG4 符号化・復号化処理方式の確立を最終的な目標として、これまでほとんど実装例のない SA-DCT の実現について述べる。

以上の目標を達成するためには、極省電力データ駆動型マルチメディアプロセッサ DDMP (Data-Driven Multimedia Processors) の利用が有望である。データ駆動型マルチメディアプロセッサは、超並列・極省電力で、1 チップ 8600MOPS(Mega Operations Per Second) の性能を持っている。本研究では、DDMP-4G というチップのフローグラフシミュレータを用いてプログラムを検証する。

MPEG4 では、VOP(Video Object Plane: フレーム) を、基本処理単位である 8×8 画素のマクロブロックに分割し、各ブロックに対して DCT を施し量子化し、量子化された DCT 係数と量子化幅を可変長符号化する。図 1.1 はマクロブロックの構造を示した図である。左上隅の画素を DC 係数と呼び、その他の画素は AC 係数とよぶ。マクロブロックに

DCT を適用すると、図 1.1 のように低周波数域から高周波数域に分解され、マクロブロックにおいて重要な情報は低周波数域に集まる。このような符号化をイントラ符号化と呼ぶ。また、符号化の対象となるマクロブロックを含む VOP に対して、時間的に隣接する別の VOP の中から、動き検出方法によって、誤差の最も小さい予測マクロブロックを検出する。予測マクロブロックへの動きを表す信号が動きベクトルであり、それに基づき、動き補償して予測マクロブロックを取得する。対象となるマクロブロックと予測マクロブロックの差分を求め、差分信号に DCT を施し、その変換係数を量子化する。これを、動きベクトルや量子化幅と共に可変長符号化する。これをインター符号化と呼ぶ。

形状符号化は、オブジェクト内外を表す信号を 2 値で表し、符号化するものである。

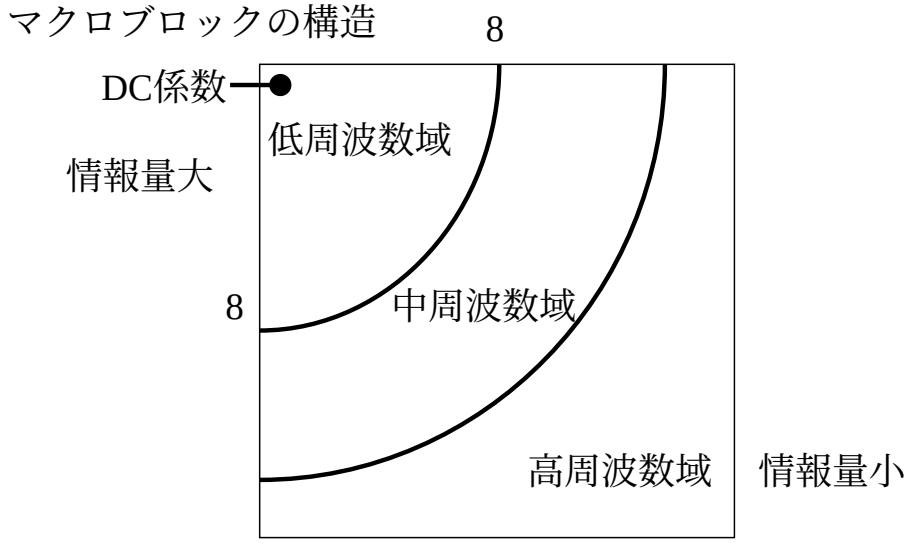


図 1.1　マクロブロックの構造

図 1.2 は VOP によるマクロブロックの種類を表している。オブジェクト符号化で VOP 内のマクロブロックは普通の符号化が行われるが、VOP 外のマクロブロックは符号化されない。また、VOP 境界部分での符号化は VOP 内部のみを符号化したほうが望ましい。そこで、境界部分のマクロブロックにおいては、MPEG4-version2 で考案された、オブジェクト内部の画素のみを DCT する、SA-DCT を適用する。[4]。SA-DCT はマクロブロックの低周波数域に画素を集め DCT を行うので、境界部分の画質の劣化を防ぐ役割もある。低周

波数域に画素を集めるのは、DC 成分に近い方が DCT 演算による情報の損失が少ないからである。つまり、低周波数域の重要な画像情報を劣化させることなく、DCT 処理できる。一般に高周波数は人間の目に判別しにくいため、符号化中で DCT の次の段階で行われる量子化で、ほとんど 0 にされてしまう。

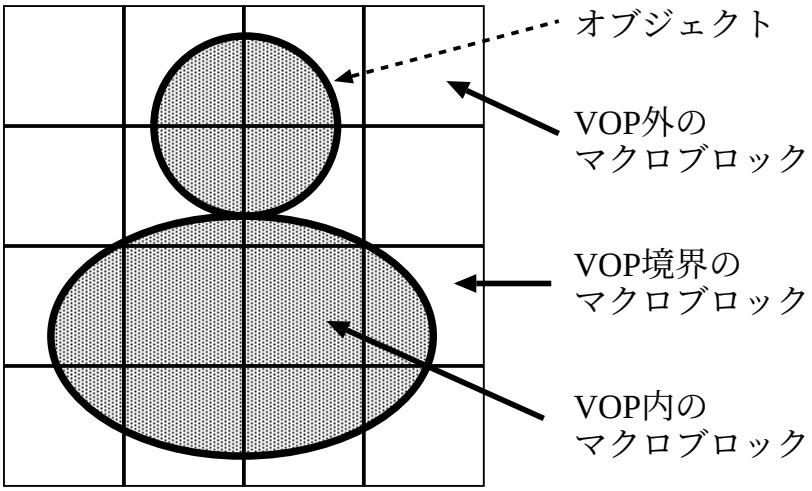


図 1.2　マクロブロックの種類

SA-DCT は、画像劣化を防ぐことはできるが、オブジェクトをマクロブロック内で左上にシフトするという 2 次元 DCT にはない操作があるため、処理に負荷がかかってしまう。

第 2 章では SA-DCT を DDMP で作成するために、SA-DCT の並列性から検証し、DDMP で実現するための要件を述べる。第 3 章では DDMP の特徴から生じる問題を解決するための方法を述べ、その一つの方法として新命令の追加を提案し、新命令を用いて、DDMP-4G 上で SA-DCT を実現したプログラム構成を提示する。第 4 章では第 3 章の提案について性能評価をする。

第 2 章

SA-DCT 処理の
データ駆動型並列実現法の要件

2.1 ブロック符号化の並列性

独立したデータについて処理を行う場合、その部分を並列に実行した方が処理レートの向上となることから、SA-DCT の高速化を図るためには SA-DCT 処理に内在する並列性を十分に抽出し、ハードウェアにより並列処理するのが望ましい。

SA-DCT は 8×8 画素からなるマクロブロック単位で処理される。図 2.1 は、そのマクロブロックの独立性を示した図である。3 つのマクロブロックはそれぞれ独立した領域を表しており、マクロブロック内の 1 ライン毎もまた互いに影響しない。各画素 (ピクセル) の情報もそれぞれ独立した値として扱うことができる。以上のことより、SA-DCT には以下の 3 つ並列性があげられる。

1. マクロブロック単位での並列性

   VOP における各マクロブロックはそれぞれ他のマクロブロックの画像情報と、独立して扱えるということである。
2. マクロブロック内のライン毎の並列性

   マクロブロック内の各ラインの処理（DCT やシフト操作）において、ライン毎にそれぞれ独立に処理することができる。
3. 各ピクセル単位の並列性

各ピクセルにおいて、ピクセルデータは互いに独立しており、それぞれのデータが独立に処理できる。

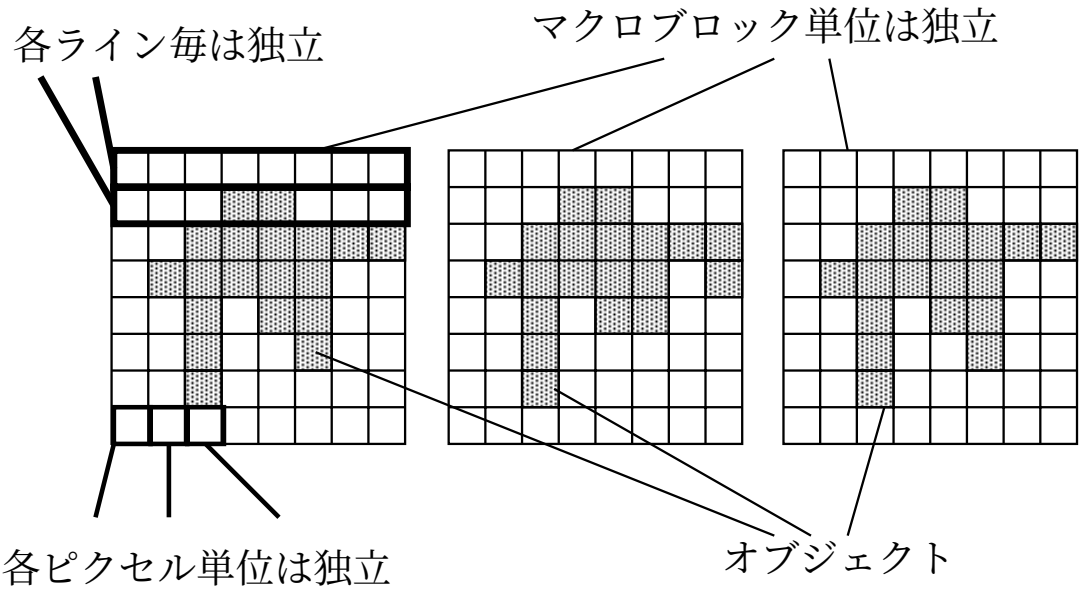


図 2.1 SA-DCT 処理の並列性

2.2 SA-DCT の概要

最初にマクロブロック単位での並列性を検討する。SA-DCT のアルゴリズムは、1 マクロブロックにおいて、オブジェクトの寄せ集めと DCT を組み合わせたものである。したがって、SA-DCT が用いられるのは境界部分のみであり、それ以外は従来の DCT と共通である。

図 2.2 は、VOP の境界部分マクロブロックにおける SA-DCT を表したものである。アルゴリズムはまず、マクロブロック内のオブジェクトのみを左方向シフトし、1 次元 DCT 処理を行う。次に、処理されたオブジェクトをさらに上方向シフトし、続いて 1 次元 DCT 処理を行う [5]。左上へのオブジェクトのシフトは、オブジェクトの画像情報を低周波数領域に集め、DCT の計算による情報の丸め込みや切り捨てを防ぎ、画像情報の損失を防ぐ役目をしている。特に、オブジェクト毎に符号化する MPEG4 では、オブジェクトの境界部分で

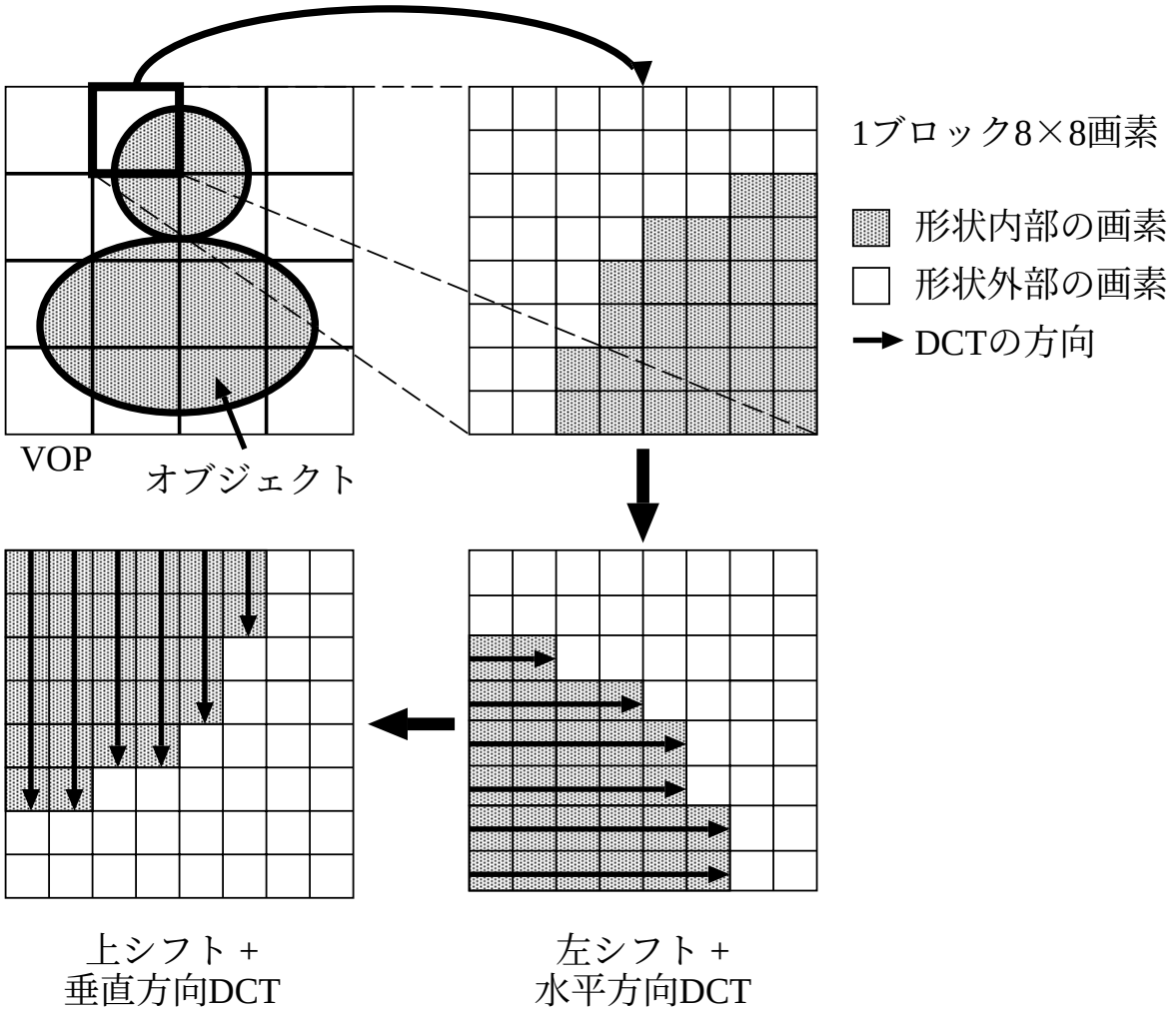


図 2.2　SA-DCT 処理アルゴリズム

その効果を発揮する。

2.3　SA-DCT の並列アルゴリズム

図 2.2 の VOP の画像は、左上の画素から順に水平方向に入力される。SA-DCT の処理アルゴリズムから、各マクロブロックの並列性より、入力と同時に水平シフトを行える。次の垂直シフトは、縦 8 ライン入力されていないと実行できないため、それまでデータを一時メモリに退避させておく必要がある。ここで、メモリに退避させてから、読み出しまでに時間が空いてしまうため、各ラインの並列性を有効に引き出して、メモリに退避させる前に 1

次元 DCT をしておく。各水平ラインの有効画素数は 1–8 まで任意であり、それぞれの基底の DCT が行われる。各ピクセル単位の並列性から、DCT 内で独立実行できる演算は並列に実行する。以上のことを考慮した上でフローグラフを作成すると、図 2.3 のようになる。

本稿で使用するアルゴリズムは、まず、複数のマクロブロックが左上から水平方向に連続して入力され、1 マクロブロック水平 1 ラインについて、オブジェクト内部の画素を左にシフトする。次に、水平 1 ラインの有効データ数の基底用 DCT、1〜8 にオブジェクト内データを分岐する。それらの中間結果をメモリに書き込み、水平方向 8 ライン分の DCT 終了後に、垂直ラインをメモリから読み出す。この出力を用いて、垂直方向 DCT も同様に繰り返す。

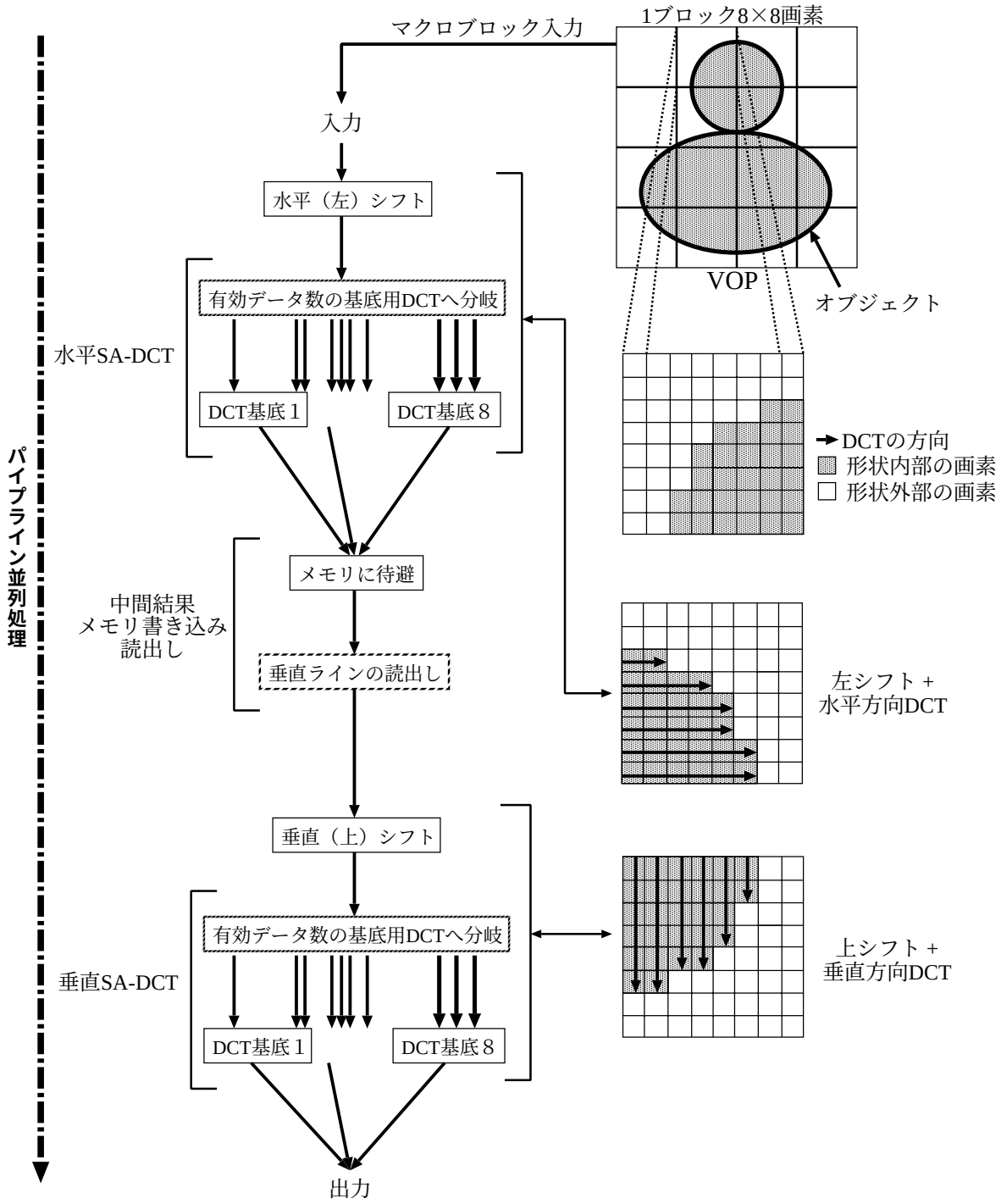


図 2.3　SA-DCT 処理フローグラフ図

2.4 DDMP とその特徴

データ駆動方式を採用したマルチプロセッサが DDMP である。DDMP は極省電力で、1 チップで 8600MOPS の性能を持っている。DDMP のメモリアクセスは制御の極小化のために、画素の 2 次元識別子 (ライン番号、ピクセル番号) で直接アドレッシング可能な構造となっている。自己タイミング型として、処理すべきデータの種類を 2 次元識別子で判別し、識別子が一致するデータのみが処理される。

DDMP の超高性能は複数のナノプロセッサを 1 チップに集積したことによる、空間並列性と、ナノプロセッサレベルでのパイプライン処理による並列性にある。図 2.4 は、DDMP の並列性を表した概念図である。x 軸は時間を表し、y 軸はパイプライン的な並列度、z 軸は空間的な並列度を表している。

DDMP のパイプラインは自己タイミング型パイプライン機構となっている。これは各々のデータが自己タイミングするため、処理に必要な部分のみを動作させることができ、省電力化できる。また、複数のパイプラインを並列に動作させ、かつ、連結して長大なパイプラインを形成できるため、柔軟な処理変更が可能であり、スループットを向上できる。

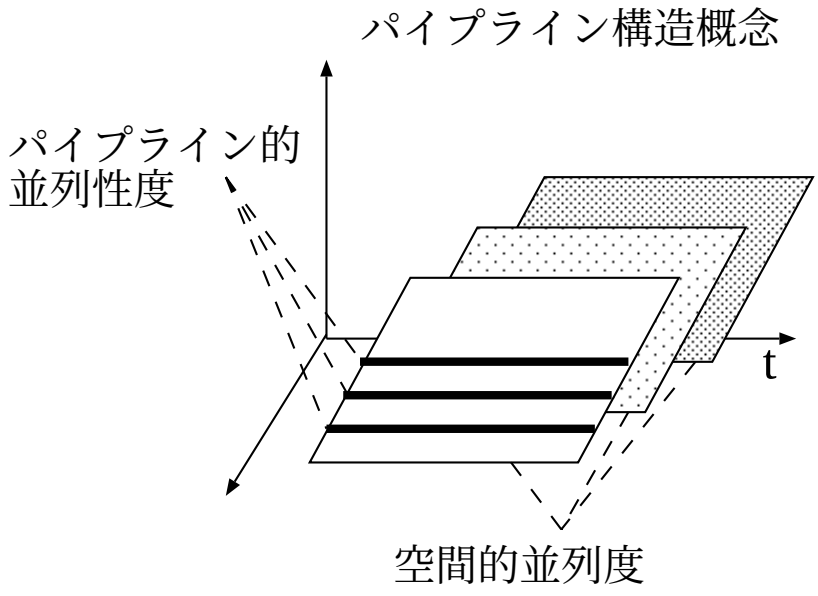


図 2.4 DDMP の並列概念図

2.5 SA-DCT 実現の課題

データ駆動型マルチメディアプロセッサ DDMP のパイプライン並列処理能力を、最大限に発揮するためには、パイプライン内のデータフロー量を最大に維持し、本来の DCT 以外のフロー制御を極小化する必要がある。

しかし、DDMP 上で図 2.3 を実現するには問題がある。現在開発されている映像処理向き DDMP-4G の命令セットでは、個々の命令がプリミティブな演算に限定されており、そのまま SA-DCT 処理を実装しても、プログラム全体の命令数が多くなりすぎ、実用的な性能を達成することができないのである。DDMP-4G では、命令数の上限が 1 ナノプロセッサ当たり 64 命令となっているためである。図 2.5 は、DDMP-4G の OCP(Operation and Control Processor) の内部構成を示している。DDMP-4G の命令はナノプロセッサ毎に異なり、DCT の演算で用いられる命令は、演算マクロ PE 中の MUL,INT(乗算演算, 整数演算) と呼ばれるナノプロセッサに格納される。

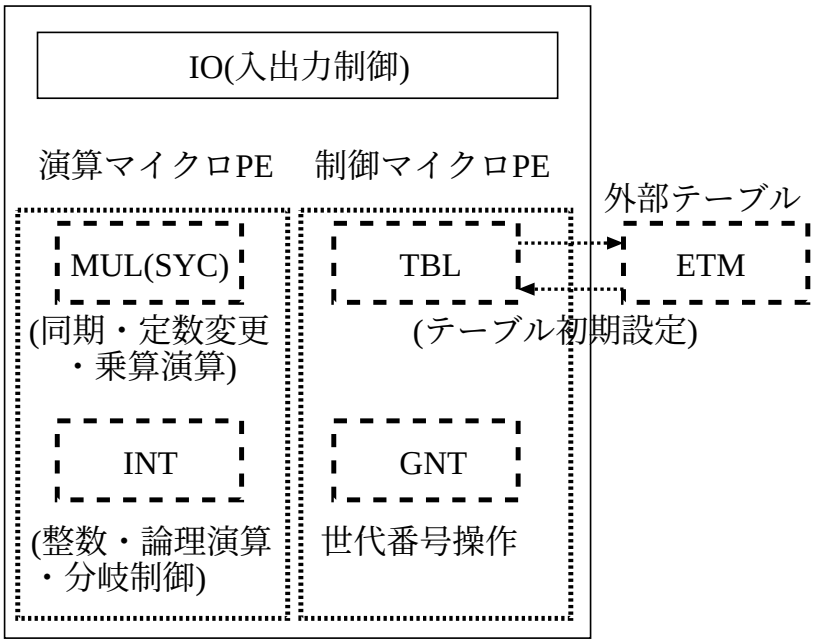


図 2.5 DDMP-4G の内部構成 (OCP)

DDMP 内のデータメモリは画素の位置情報で直接アクセス可能なアドレス空間を採用しており、データトークンに付加された 2 次元識別子 (ライン番号、ピクセル番号) をキーに

して、ビデオメモリに容易にアクセスできる命令セットが提供されている。しかし、1 回のメモリアクセスで高々1 つのデータしか参照/更新できないため、SA-DCT における画素のシフト操作のように、不規則なメモリアクセスを行うプログラムは、非常に命令数が多くなる。さらに、単純な分岐命令しかないため、動的なデータ依存関係の表現も命令数が多くなる。そのため、従来の DCT 処理では不要なシフト操作部分が、符号化・復号化処理全体のボトルネックとなってしまう。

高速 DCT は有効画素をシフトする操作がないために、SA-DCT より命令数が少なくてよい。高速 DCT とは、普通の DCT のアルゴリズムを変換させて、乗算加算回数を $(N_a N_2)^2$ 回から $N_1^2 N_2 + N1 N_2^2$ 回に削減できるアルゴリズムである [6]。高速 DCT と SA-DCT を比較すると、境界部分での画質の劣化減少として SA-DCT が優れているが、処理アルゴリズムの複雑性から、高速 DCT の方が圧倒的に命令数が少なく、高速に DCT 処理が可能となる [4]。

SA-DCT の処理能力を高速 DCT の処理能力に近づけるためには、DCT 処理以外での命令数の削減が必須となる。

つまり、本来の DCT 処理にない部分でのボトルネックを解消することが、課題となる。

第 3 章

SA-DCT 処理データ駆動並列の最適化

3.1 ボトルネックの根拠

本章では、現行の DDMP-4G で SA-DCT 処理を実装した場合に、ボトルネックとなってしまうシフト操作やメモリ参照の部分を、解消するための方法を示す。

SA-DCT を DDMP-4G で実装した場合命令数が多すぎて、現在の DDMP-4G ではハードウェア的な命令数制約によりシミュレートできないため、実測値は出せない。

表 3.1 命令数割合の比較 (高速 DCT と SA-DCT との比較)

DCT 基底数	1	2	3	4	5	6	7	8
高速 DCT	1	1	1	1	1	1	1	1
SA-DCT	6.3	6.3	6.6	6.9	7.2	8.1	9.3	8.4

高速 DCT と SA-DCT の命令数の割合で比較すると、表 3.1 のようになる。SA-DCT は、有効画素 (オブジェクト内の画素) 数により、基底の DCT が変化するため、DCT 部分において、SA-DCT の方が命令数が少ない場合があるが、DCT の前処理として行うシフト操作や、メモリ操作の分だけ命令数が多くなってしまう。そのため、全体としては SA-DCT の方が 6 倍から 10 倍の命令数となってしまう。第 2.5 節で示したように、これは DDMP-4G の命令数制限を越えてしまう。

3.2　ボトルネックの解消法

前に述べた根拠より、SA-DCT でボトルネックとなる部分を以下に示す。

(a)　シフト操作

1 つ目のボトルネックは、水平方向シフト垂直方向シフトである。この操作は SA-DCT がもつ特有の処理であり、この部分で遅延は SA-DCT のアルゴリズムから生じている。

(b)　分岐操作

2 つ目は分岐部分である。これは DDMP の特徴から生じるもので、現在の DDMP には単純な分岐の機能しかないため、3 つ以上の場所に分岐させたい場合、単純 2 分岐命令を繋げて多段にしなければならない。多段にすればデータが通過する距離が長くなってしまうので、指定の位置に分岐するまでに遅延が生じる。

(c)　メモリ操作

3 つ目はメモリに待避しておいたデータを垂直方向に読み出す部分である。これも DDMP の特徴から生じるもので、1 つのデータで 1 つのアドレスしか参照できないために、複数の有効データを参照することはできない。そのため、1 つのデータを複製し、その複製データで一つづつメモリアクセスしなければならない。また、メモリアクセスは and や or などの普通の命令に比べ、参照・更新に時間がかかるため、さらに遅延が増大してしまう。

これらのボトルネックの解消法として、以下のものがあげられる。

1. 処理アルゴリズムの変更
2. ハードウェアによるサポート

3.2.1　処理アルゴリズムの変更

1 つ目の解消法は、図 2.3 のアルゴリズム自体の変更である。(a) の部分は、シフトアルゴリズムに工夫を凝らすことにより、命令数を削減することができる。DDMP では、世代という概念より処理対象となるデータが識別される。DDMP-4G では、有効画素の位置情報を世代 (ピクセル番号、ライン番号) としているが、画素のシフトは、この世代を操作することになり、非常に重たい処理となっている。DDMP-4G に用意されている命令セットを用いると、数パターンのシフトアルゴリズムが表現可能であり、その中で最適となるアルゴリズムを選定すれば、ある程度の性能は期待できる。

しかし、メモリ操作の部分は、1 データで参照・更新できるデータは高々一つとなっている、DDMP 特有の問題はそのまま残るため、アルゴリズムの変更だけでは十分な実行速度は得られず、ボトルネックを十分解消できない。

3.2.2　ハードウェアによるサポート

アルゴリズムの最適化だけでは、ボトルネックを解消できないとなると、遅延の起こる部分をハードウェアで補うことが考えられる。

SA-DCT 中で処理が複雑なためにボトルネックとなっているのは、有効画素数の基底に応じた DCT へと分岐させる分岐操作と、水平ライン DCT 後のデータを一時メモリに書き込み、水平 8 ラインが終了した時点でメモリから読出すという、1 個のデータから複数のデータを参照するメモリ操作である。

この部分を新しい分岐操作とメモリ読み出しの機能を、それぞれ一度に行えるようなハードウェア機構を追加する。これにより、多段かつ多命令になってしまう分岐操作、メモリ操作を 1 命令で実行可能となり、(b),(c) のボトルネックが解消できる。

3.3 追加する新命令

前節までの検討を基に DDMP 上でソフトウェアによる SA-DCT 機能を実現するために、簡単なハードウェア機構により実装可能な新命令を提案する。

今回の研究ではハードウェアの設計・実装は行わないが、最終的には設計実現可能とすることを視野に入れており、今回は、ハードウェアが実装可能であると仮定した場合に、性能の向上性を定性的に示す。

3.3.1 メモリ連続参照命令 (h-read, v-read)

(c) を解消するためには、次に示す h-read, v-read を追加する。

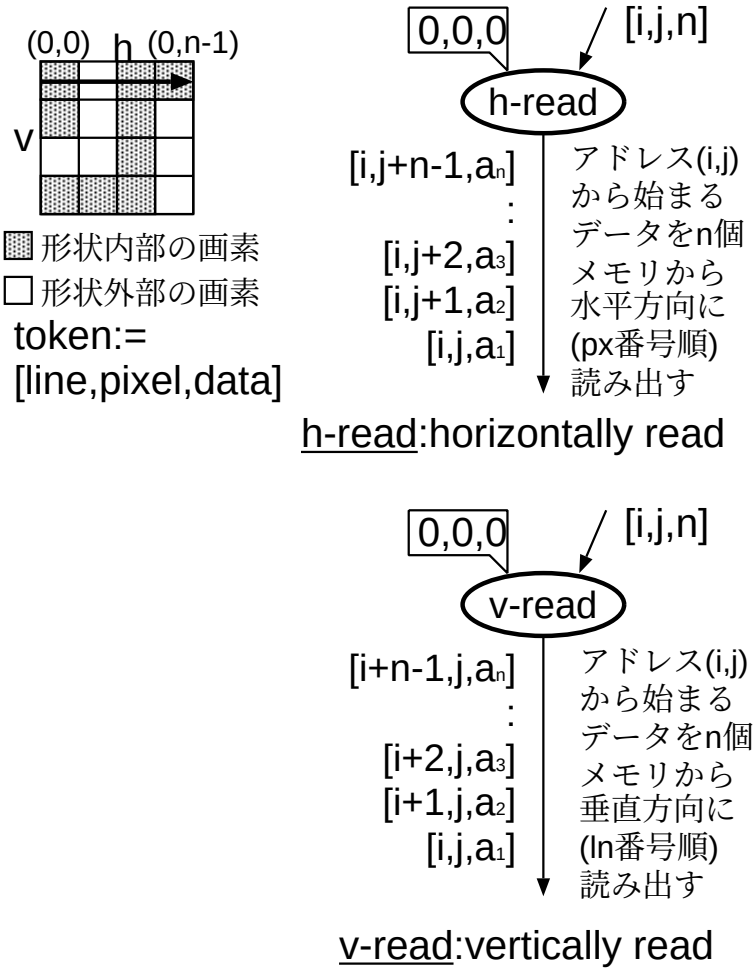


図 3.1 メモリ連続参照命令

- h-read

  図 3.1 は、右入力データの世代番号 (i=ライン番号, j=ピクセル番号) と、左入力 (定数) を加えた値のアドレスから始まるメモリデータを、n=データ個だけピクセル番号順に読出す。

  (左入力データ + j) = テーブルアドレス ⇒ n 個出力 (ピクセル番号順)

- v-read

  v-read も同様に、右入力データの世代番号と、左入力を加えた値のアドレスから始まるメモリデータを、n=データ個だけライン番号順に読み出す。

  (左入力データ + i) = テーブルアドレス ⇒ n 個出力 (ライン番号順)

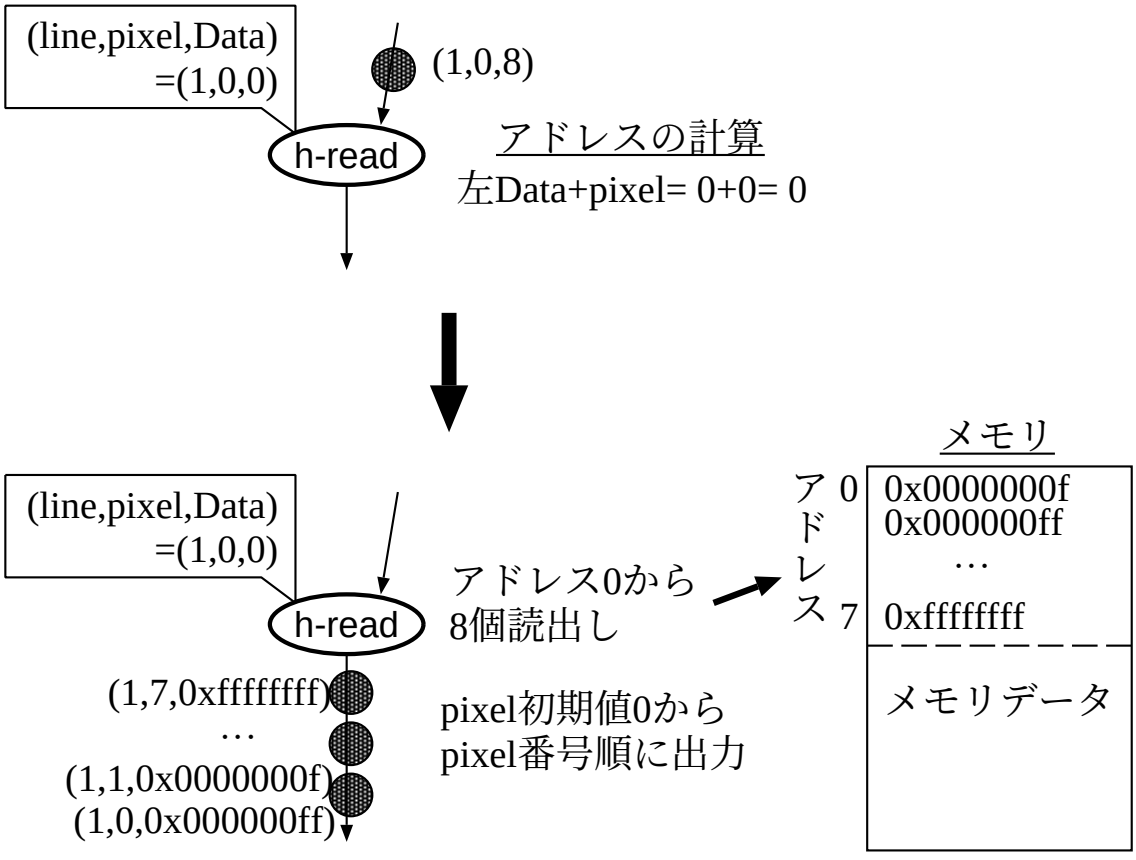


図 3.2 h-rad の使用例

例えば、図 3.2 のように、左入力データが (1,0,0)=(ライン番号, ピクセル番号, データ) で、右入力データが (a,0,8) の場合の `h-read` は、ピクセル番号 0〜7 が指定する 8 つのデータがメモリから参照されることになる。

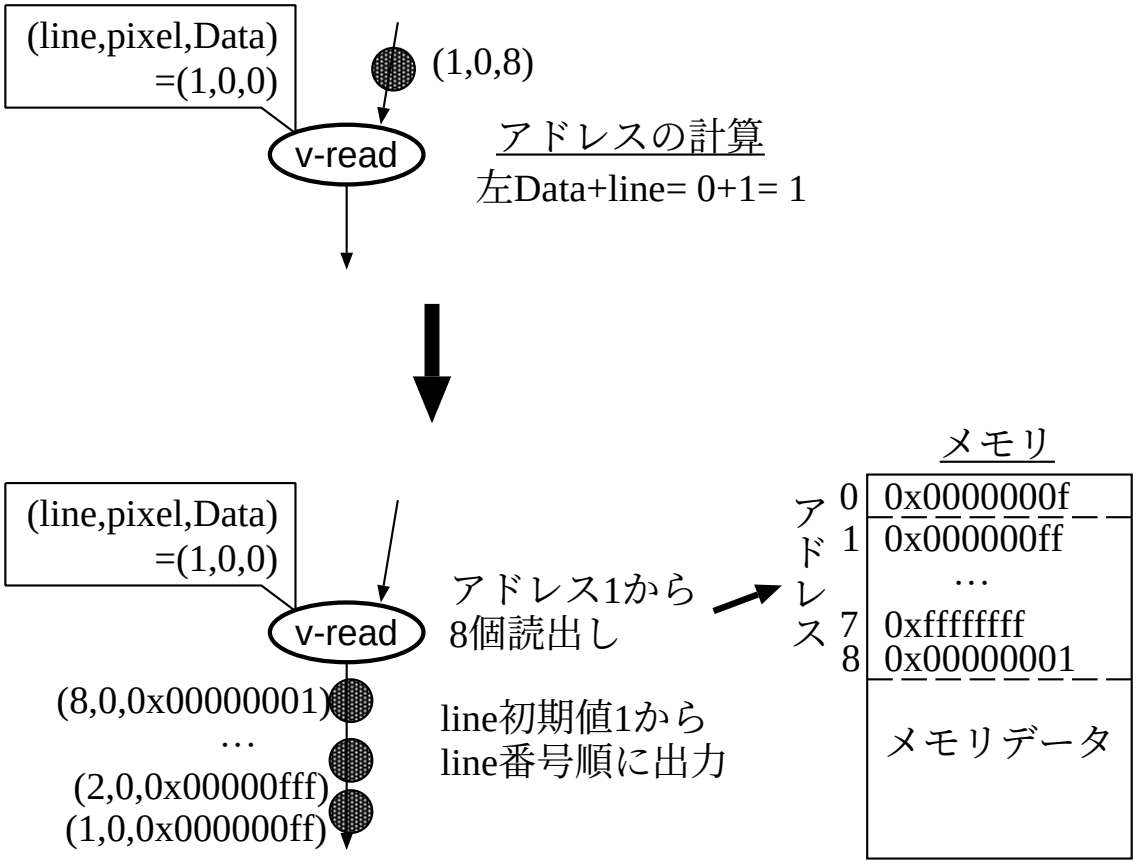


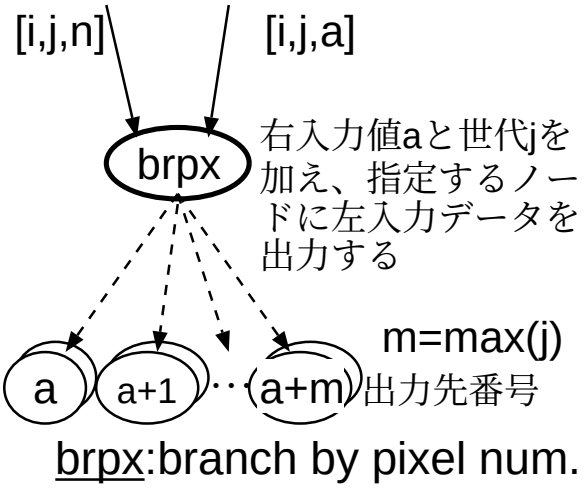
図 3.3　v-read の使用例

図 3.3 の `v-read` はライン番号 0〜7 が指定する 8 つのデータが参照される。

これを用いて、マクロブロック内の 1 行 (列) のデータを一度に参照することができる。

3.3.2 2 フェーズ多分岐命令 (brpx, brln)

(b) を解消するために、次に示す brpx, brln の 2 命令を追加する。



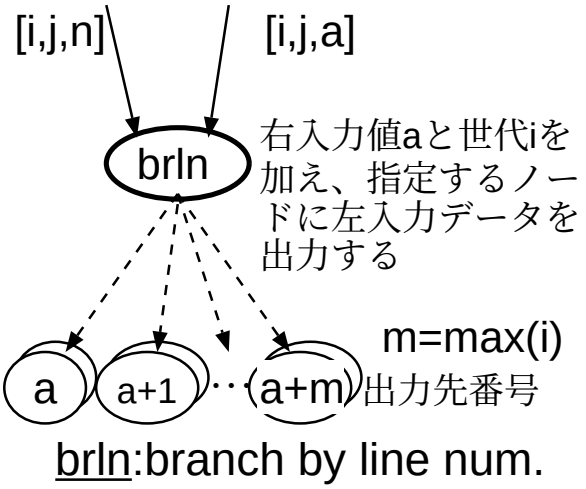


図 3.4 2 フェーズ多分岐命令

- brpx

  図 3.4 は、右入力データ a に世代番号 (ピクセル番号 j) を加えた値で、左入力データ n を指定先ノードに分岐させて出力する。

  (a + j) = 指定先ノード数 ⇒ n を指定先ノードに分岐

- brln

  brln も同様に、右入力データ a に世代番号 (ライン番号 i) を加えた値で、左入力データ n を指定先ノードに分岐させて出力する。

  (a + i) = 指定先ノード数 ⇒ n を指定先ノードに分岐

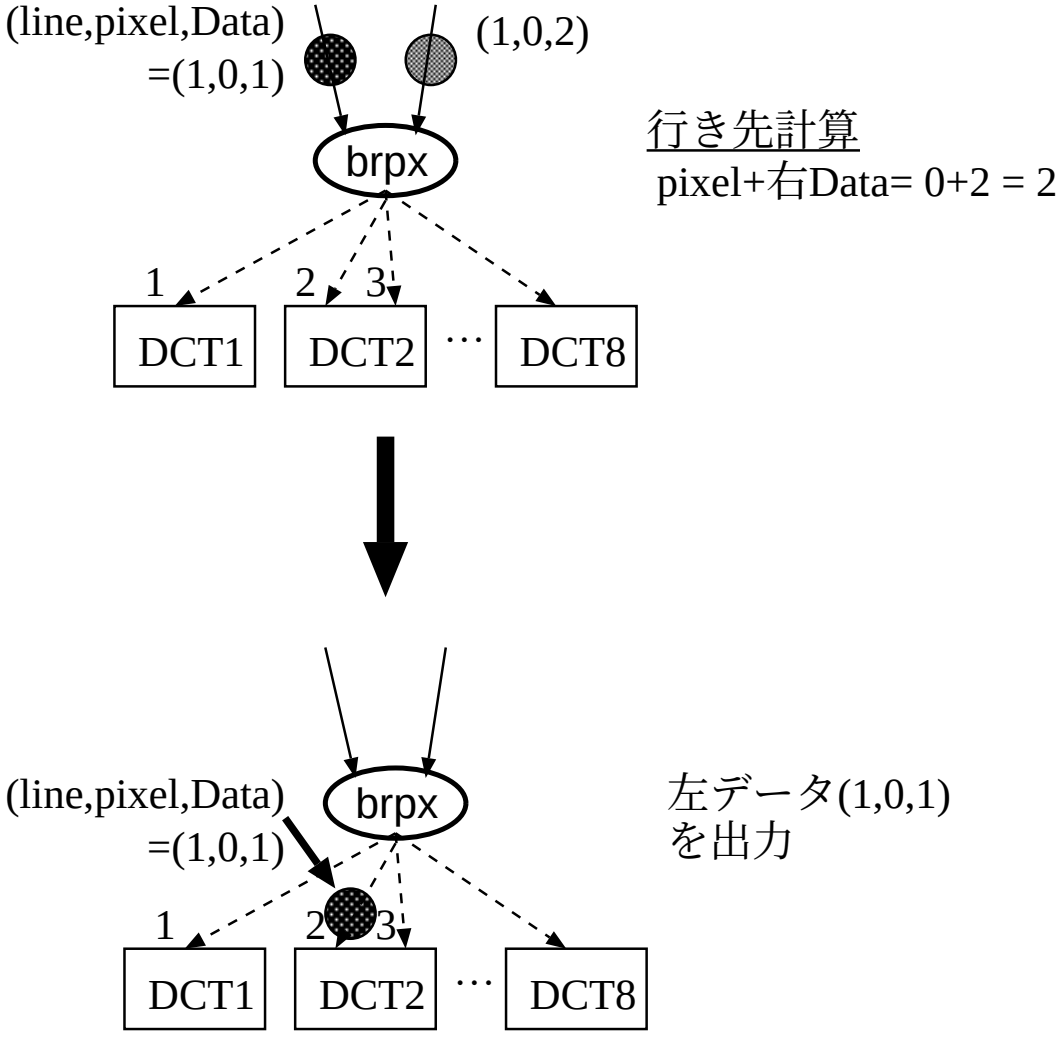


図 3.5 brpx の使用例

例えば、図 3.5 のように、左入力データが (1,0,1)=(ライン番号, ピクセル番号, データ) で、右入力データが (1,0,2) の場合の brpx は、右データの 2 とピクセル番号の 0 を加えた、指定先番号 2 のノードに左データを分岐させる。

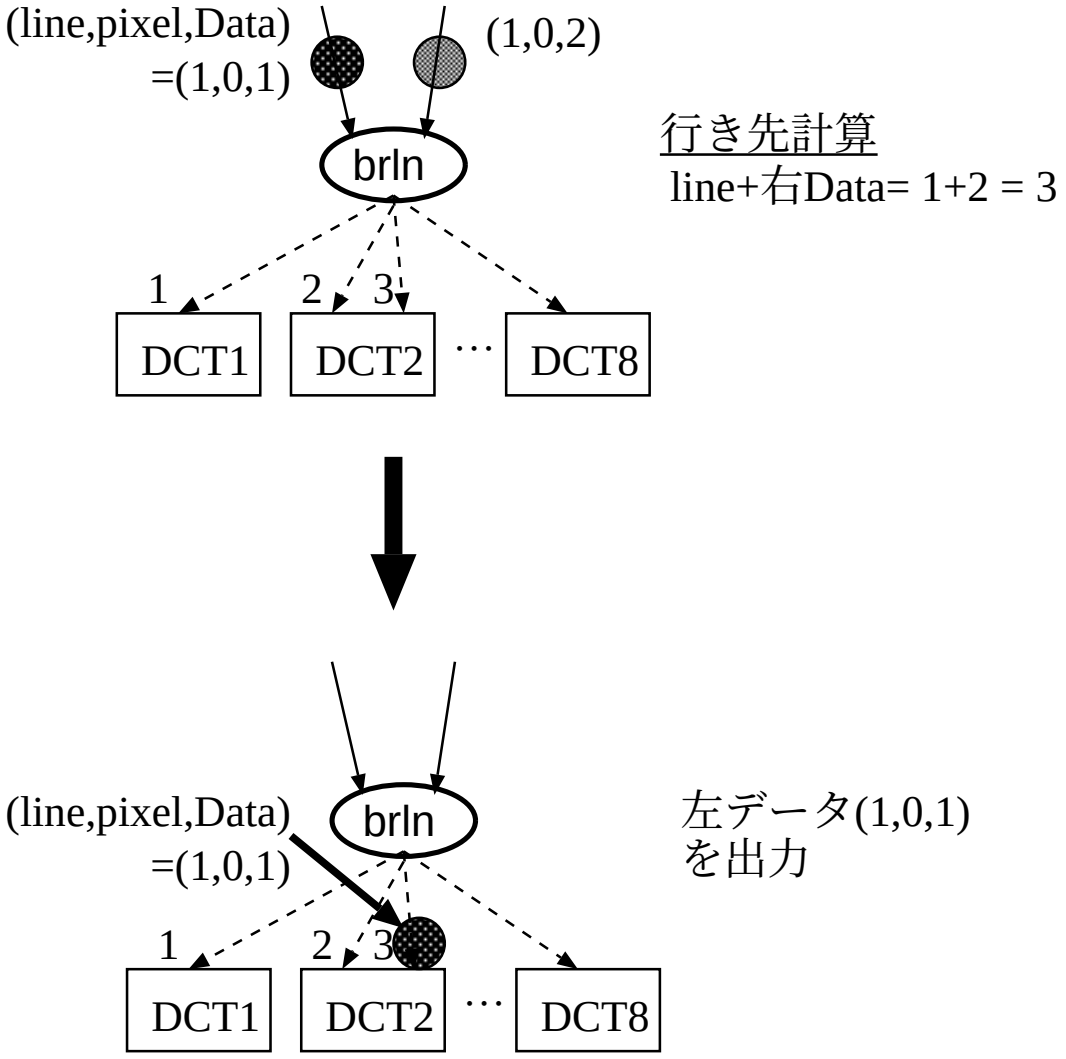


図 3.6　brln の使用例

図 3.6 の brln は、右データの 2 とライン番号の 1 を加えた、指定先番号 3 のノードに左データを分岐させる。

これを用いて、シフトさせた有効データを基底の DCT へ多分岐させることができる。

3.3.3 新命令の柔軟性

本研究は、プログラム可能な機能の追加を目標としているので、追加する 4 つの新命令は、SA-DCT 用に特化したハードウェア機構とするのではなく、応用できるように柔軟性を持たせてある。

例えば、メモリ連続参照命令の場合、左入力値を定数としないで、1 個のデータを入力させることで、任意なメモリデータを取得することができる。これは、他の画像処理アルゴリズムにも有用である。

2 フェーズ多分岐命令の場合、右入力のデータを定数値 0 としておくと、左入力の世代番号 (`brpx` の場合はピクセル番号、`brln` の場合はライン番号) のみを分岐計算の対象とできる、従来の分岐の自然な拡張となっている。

3.4 SA-DCT の作成

図 3.7 は、追加した新命令を用いた SA-DCT のフローグラフである。メモリ連続参照命令は、垂直ラインの画素を読み出す部分で使用し、2 フェーズ多分岐命令は各基底の DCT へ分岐する部分で使用する。その他の部分の処理は、2 フェーズ多分岐命令のための、データ数カウント部とそれぞれの基底の DCT 部分とに分けられる。

まず画像が入力され、有効データ数をカウントする。次にカウントされたデータ数だけ、そのカウント数基底の DCT へと 2 フェーズ多分岐命令 (`brpx`) で分岐する。基底の DCT で 1 次元 DCT をし、一度メモリに書き込みを行う。これは、データが水平方向順に入力されるので、次の垂直方向 DCT を行える状態（縦 1 ラインの画像が総てそろった場合）になるまで、データをメモリ格納しておかないと、待ち合わせのメモリが消費されてしまうからである。縦 1 ラインがそろった状態になると、メモリ連続参照命令 (`v-read`) を用いて、マクロブロックの縦 1 列の画像情報をメモリから読み出す。さらに各基底の DCT への分岐 1 次元 DCT と繰り返される。

これらの一連の処理をパイプライン並列で処理するのに、実現可能な命令数にできた。

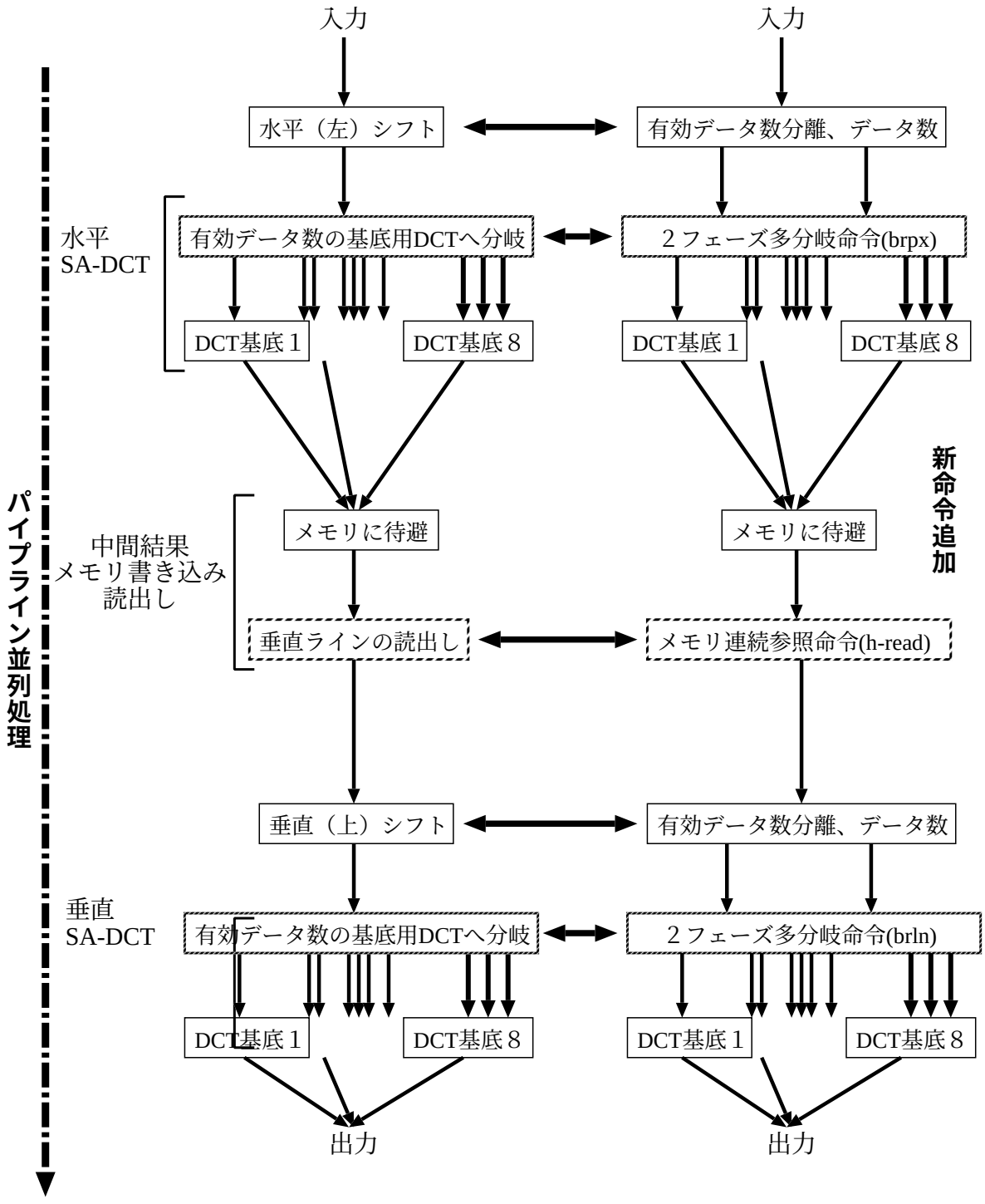


図 3.7 SA-DCT 処理フローグラフ図 (新命令追加)

第 4 章

性能評価

4.1 シミュレータの原理

本研究のシミュレーションは、DDMP-4G のフローグラフシミュレータで行う。このシミュレータは DDMP チップの構成を基にした、フローグラフシミュレーションツールである。プログラムは、データの流れるフローグラフを作成し、シミュレータがこれを解析して入力データを演算する。入力データは世代番号とエントリ、データを持つパケットデータであり、本研究では、1 パケットのデータを 1 画素のデータとしている。これは、DDMP-4G のパケットが 12bit のデータまでしか扱えないためである。

4.2 シミュレーション手法

シミュレーション入力データの選定方式としては以下のことがいえる。

- オブジェクトパターンの相違に、性能は無関係

  1 マクロブロックにおいて、世代情報をオブジェクトの位置情報としているため、有効画素値をシフトすることは世代情報を操作することとなる。DDMP はデータ駆動方式であるため、処理すべきデータかどうかで判断している。さらに、どの位置に有効画素が存在していても、処理毎に位置情報 (世代情報) が変わるため、入力する有効画素値の位置による性能の変化はない。
- マクロブロック内でのオブジェクトの占有率に依存

  有効画素値の位置による性能の変化はないが、1 マクロブロックにおいて、有効画素値

がどれだけ存在するかという占有率によって、データの通るフローグラフ中の場所 (基底の DCT) が異なり、性能が決まる。

- 入力データパケット数が最大時の性能を評価

  DDMP は処理の終了した部分については実行しないようになっているため、プログラム中の負荷は実行されている命令だけに限定されてしまう。入力データ量が少ないと、本当の性能を評価したことにはならない。そこで、データ数をプログラム中の命令実行段数より多く取る必要がある。つまり、データが入力されて、最初の 1 画素が出力されるまでは、データを入力し続けなければならない。

以上のことから、入力パケットの選定として、有効データの位置 (世代番号) とは無関係に、占有率の違いによる入力データを用意する。入力データ数は、プログラム中総てにデータが流れている状態にするために、命令実行段数 (速度) より多いデータ数を入力しなければならない。このデータ数はプログラム実行速度と、入力データの投入速度によって決定する。

例えば、64 画素数 (8×8) の、16 マクロブロック (4×4) とすると、入力パケット数は以下の式で算出される、1024 パケットとなる。

$$64(\text{1 マクロブロック中の画素数}) \times 16(\text{マクロブロック数}) = 1024(\text{入力パケット数})$$

また、パケットの投入間隔が 30 サイクルとすると、

$$30(\text{入力データ投入間隔}) \times 1024(\text{入力データ数}) = 30720 \text{ サイクル}$$

となり、最初に出力されるデータが、この値より早くなければならない。

マクロブロック内のオブジェクト占有率は、0,4,8,16,32,64 の 7 パターンで検証する。

4.3 測定方式

測定式は以下のものとする。

(1) 命令数:

プログラム中で、データが通過する命令数 (個)

(2) スループット:

$$\frac{1}{\text{処理はじめて出力されるまでのサイクル数} \times \text{時間}}(DCT\ \text{処理}/s)$$

$$\frac{\text{スループット}}{\text{1 フレームのマクロブロック数}}(fps)$$

(3) レスポンスタイム:

1 マクロブロックデータが入力されて出力されるまでの時間 (s)

(1) は、各命令に実行時間が割り振られており、命令実行時間の長い命令を多用すると、データフローレートが低下する。このため、命令数がプログラムの実行速度を左右するといえ、本研究のプログラム評価指標の一つとする。

(2) は、コンピュータシステムの処理能力を測る評価指標であり、本研究では、一定時間内に SA-DCT 演算が、いくつのマクロブロックにおいて実行できるかを測定する。

(3) も性能評価指標の一つで、データが入力されてから、出力されるまでの時間のことである。本研究では、1 マクロブロックが入力されてから、SA-DCT 処理され総て出力されるまでの時間を測定する。

これらの測定式は、入力データがプログラムで処理されて出力され、それがある一定の間隔になっている状態の部分、つまりプログラム中で、定常状態になった時の出力の、1 マクロブロックについて適用する。

4.4 性能比較

旧方式の SA-DCT は命令数が多すぎ、現在の DDMP-4G の命令数制約によりシミュレートできない。実測値は出せないため今回の比較は命令数で行う。

4.4.1 旧方式 SA-DCT との比較

旧方式 SA-DCT とは、新命令を使わずに SA-DCT を DDMP-4G で実装した場合の SA-DCT のことである。

命令数の割合は表 4.1 のようになり、これは新命令追加の SA-DCT の、オブジェクト占有率 0 の場合を 1 とした時の割合である。新命令追加 SA-DCT 処理の方が、シフト操作、メモリ読み出しが 1 命令で良いため、明らかに命令数の割合が少ないと考えられる。

しかし、各基底の DCT へ分岐するために、有効画素数をカウントしなければならず、圧倒的な差とはなっていない。

表 4.1 命令数割合の比較 (旧方式 SA-DCT との比較)

オブジェクト占有率	旧方式 SA-DCT	新命令追加 SA-DCT
0	1.27	1
4	1.32	1.04
8	1.36	1.08
16	1.49	1.21
32	1.55	1.27
64	1.66	1.38

4.4.2 高速 DCT との比較

高速 DCT とは、DCT 処理を高速に処理するためのアルゴリズムで、普通に DCT の式をプログラム化するよりも、はるかに命令数が少なく、処理速度も速い。

高速 DCT と SA-DCT の命令数の割合は表 4.2 となり、これは高速 DCT の命令数を 1 とした時の割合である。高速 DCT はオブジェクト占有率にかかわらず、総ての画素について DCT を行うので、割合は総て 1 となる。それに対して SA-DCT は、オブジェクトの占有率によって、任意にフローする場所が違い、命令数が変化するため命令数は多くなっている。

表 4.2 命令数割合の比較 (高速 DCT との比較)

オブジェクト占有率	高速 DCT	新命令追加 SA-DCT
0	1	2.05
4	1	2.14
8	1	2.23
16	1	2.49
32	1	2.62
64	1	2.83

第 5 章

結論

本研究では、柔軟性のある MPEG4 画像符号化の実現として、SA-DCT のデータ駆動型並列実現法について述べた。

第 2 章では、SA-DCT 処理の概要と内在する並列性を述べ、その具体的な並列アルゴリズムをモデル化した。そしてその並列性から、DDMP 処理方式に基づいた SA-DCT アルゴリズムを検証し、処理の高速化を測る上で、フローグラフ化の問題点を取り上げた。

第 3 章では、SA-DCT を DDMP で実装する場合の最適化法を検証し、簡単なハードウェア機構の追加による高速化を提案した。いかに汎用性のあるハードウェア機構を追加するのか、ということが重要である。新命令を追加するにあたり、SA-DCT だけに特化したものであると、柔軟な方式とはいえないからである。

第 4 章では、提案した新命令を用いた SA-DCT の性能評価基準をいくつか述べ、その中で、現在可能な評価として、命令数を比較した。比較対象は、新命令を用いない場合の SA-DCT と、高速 DCT とで行った。

本研究の結果としては、SA-DCT プログラム中で、オブジェクト占有率に応じてデータが通過する基底の DCT が変わり、オブジェクト数が少ないほど通過する命令実行段数が少ない。つまり、オブジェクト占有率が低いときは、シフトによる遅延が DCT 処理で解消されるため、性能がほぼ同等になる場合がある。しかし、有効画素数のカウント部分を、更に最適化しないと、新命令の追加によって命令数は減ったものの、有用な性能を得るまでには至らない。

今後の課題として、現在までに、Java によるデータ駆動型マルチプロセッサシミュレータをほぼ完成させたので、今後はこのシミュレータを用いて、新命令追加による SA-DCT

処理の詳細な性能評価を実施する予定である。

他にも、命令数から見て、高速 DCT と比較すると、従来の SA-DCT からすると減ったものの、十分な性能を得ているとはいえない。その原因となっている、有効画素数のカウント部分を最適化する必要がある。将来的には、今回提案した新命令をハードウェアで作成し、実際に使える命令として DDMP チップに追加する予定である。

また、今回の新命令作成にあたり、ハードウェア機構に汎用性を持たせているため、SA-DCT 処理だけにとどまらず、量子化、動き補償などをはじめとした、幅広い分野への応用が考えられる。

謝辞

本研究に於いて、懇切なるご指導、ご鞭撻を賜った高知工科大学の 岩田 誠 助教授に、心より感謝の意を表します。

本研究の基礎としている自己タイミング型データ駆動マルチプロセッサの考え方を提唱され、様々な御示唆を賜った 寺田 浩詔教授 に心より感謝の意を表します。

本研究を進めるにあたり、適切な御助言、ご指導を賜った 大森 洋一 助手に、心より感謝の意を表します。

本研究を進めるにあたり、シミュレータ環境開発に携わった、パシフィックソフトウェア開発株式会社、シャープ株式会社の開発担当者の方々に、感謝の意を表します。

論文執筆にあたり、日頃から温かいご支援を頂いた、 細美 俊彦 氏に、感謝の意を表します。

日頃からご支援頂いた情報システム工学科岩田研究室の方々、森川 大智 氏、前田 庸亮 氏、森安 亮 氏、別役 宣奉 氏、古家 俊之 氏、假屋 文彦 氏に感謝の意を表します。

参考文献

[1] H. Terada, S. Miyata, and M. Iwata, "DDMP's: self-timed super-pipelined data-driven multimedia processors," *Proc. of IEEE*, 87(2), 282–296 (1999).

[2] 三木弼一 編, "MPEG-4 のすべて," 工業調査会 (1999).

[3] K. R. Rao, and J. J. Hwang 著, 安田 浩, 藤原 洋 監訳, "デジタル放送・インターネットのための情報圧縮技術," 共立出版 (1999).

[4] 樫田 祐介, 矢ヶ崎 陽一, 渡辺 敏明, 甲藤 二郎, "MPEG-4 規格の新展開," 映像情報メディア学会誌, 53(4), 485–491 (1999).

[5] 黒田 涼, 藤田 玄, 尾上 孝雄, 白川 功, "MPEG-4 向け省面積 SA-DCT の VLSI 化計画," *TECHNICAL REPORT OF IEICE.*, 2000(35), (2000).

[6] W. H. Chen, C. H. Smith, and S. C. Fralick, "A Fast Computational Algorithm for the Discrete Cosine Transform," *IEEE Trans. Commun.*, 25(9), 1004–1009 (1997).